**Practical no 9**

**/\***

**🔘PROBLEM STATEMENT:-**

**A Dictionary stores keywords & its meanings. Provide facility**

**for adding new keywords, deleting keywords, updating valuesof any**

**entry. Provide facility to display whole data sorted in ascending/**

**Descending order. Also find how many maximum comparisons may**

**require for finding any keyword. Use Height balance tree and find the**

**complexity for finding a keyword**

**\*/**

#include<iostream>

#include<stdlib.h>

#include<string.h>

using namespace std;

class avlnode

{

public:

char keyword[20];

char meaning[30];

int ht;

avlnode \*left;

avlnode \*right;

avlnode()

{

left=right=NULL;

}

};

class dictionary

{

public: avlnode \*root;

void addkeyword();

avlnode\* place\_keyword(avlnode\*,avlnode\*);

avlnode \*LL(avlnode\*);

avlnode \*RR(avlnode\*);

avlnode \*RL(avlnode\*);

avlnode \*LR(avlnode\*);

avlnode \*rotateright(avlnode\*);

avlnode \*rotateleft(avlnode\*);

int balance(avlnode \*);

int height(avlnode\*);

void display\_asc();

void inorder(avlnode \*);

void display\_dsc();

void rtorder(avlnode \*);

void search\_keyword();

void update\_keyword();

avlnode \*avlsearch(avlnode\*,char[]);

dictionary()

{

root=NULL;

}

};

void dictionary::addkeyword()

{

int no,i;

avlnode \*temp;

cout<<"\nEnter no of Keywords : ";

cin>>no;

for(i=0;i<no;i++)

{

temp=new avlnode();

cout<<"\nEnter Keyword : ";

cin>>temp->keyword;

cout<<"\nEnter meaning : ";

cin>>temp->meaning;

root=place\_keyword(root,temp);

}

}

avlnode\* dictionary::place\_keyword(avlnode\* r,avlnode \*temp)

{

if(r==NULL)

{

r=temp;

}

else if(strcmp(temp->keyword,r->keyword)<0)

{

r->left=place\_keyword(r->left,temp);

if(balance(r)==2)

{

if(strcmp(temp->keyword,r->left->keyword)<0)

{

r=LL(r);

}

else

{

r=LR(r);

}

}

}

else if(strcmp(temp->keyword,r->keyword)>0)

{

r->right=place\_keyword(r->right,temp);

if(balance(r)==-2)

{

if(strcmp(temp->keyword,r->right->keyword)>0)

{

r=RR(r);

}

else

{

r=RL(r);

}

}

}

r->ht=height(r);

return r;

}

avlnode\* dictionary::LL(avlnode \*temp)

{

temp=rotateright(temp);

return temp;

}

avlnode\* dictionary::RR(avlnode \*temp)

{

temp=rotateleft(temp);

return temp;

}

avlnode\* dictionary::RL(avlnode \*temp)

{

temp->right=rotateright(temp->right);

temp=rotateleft(temp);

return temp;

}

avlnode\* dictionary::LR(avlnode \*temp)

{

temp->left=rotateleft(temp->left);

temp=rotateright(temp);

return temp;

}

avlnode\* dictionary::rotateright(avlnode \*temp)

{

avlnode \*y;

y=temp->left;

temp->left=y->right;

y->right=temp;

temp->ht=height(temp);

y->ht=height(y);

return y;

}

avlnode\* dictionary::rotateleft(avlnode \*temp)

{

avlnode \*y;

y=temp->right;

temp->right=y->left;

y->left=temp;

temp->ht=height(temp);

y->ht=height(y);

return y;

}

int dictionary::balance(avlnode \*temp)

{

int lh,rh;

if(temp==NULL)

return(0);

if(temp->left==NULL)

lh=0;

else

lh=(temp->left->ht)+1;

if(temp->right==NULL)

rh=0;

else

rh=(temp->right->ht)+1;

return(lh-rh);

}

int dictionary::height(avlnode \*temp)

{

int lh,rh;

if(temp==NULL)

return(0);

if(temp->left==NULL)

lh=0;

else

lh=(temp->left->ht)+1;

if(temp->right==NULL)

rh=0;

else

rh=(temp->right->ht)+1;

if(lh>rh)

return (lh);

else

return(rh);

}

void dictionary::display\_asc()

{

cout<<"\n\n Keyword Meaning";

cout<<"\n---------------------------------";

inorder(root);

}

void dictionary::inorder(avlnode \*r)

{

if(r!=NULL)

{

inorder(r->left);

cout<<"\n"<<r->keyword<<" "<<r->meaning;

inorder(r->right);

}

}

void dictionary::display\_dsc()

{

cout<<"Keyword"<<" "<<"Meaning";

cout<<"\n---------------------------------";

rtorder(root);

}

void dictionary::rtorder(avlnode \*temp)

{

if(temp!=NULL)

{

rtorder(temp->right);

cout<<"\n"<<temp->keyword<<" "<<temp->meaning;

rtorder(temp->left);

}

}

void dictionary::search\_keyword()

{

char targetkey[20];

avlnode \*temp;

cout<<"\nEnter Keyword To Be Searched :";

cin>>targetkey;

temp=avlsearch(root,targetkey);

if(temp==NULL)

{

cout<<"\nKeyword Is Not Present In The Dictionary..!!..";

}

else

{

cout<<"\n"<<temp->keyword<<" "<<temp->meaning;

}

}

void dictionary::update\_keyword()

{

char targetkey[20];

avlnode \*temp;

cout<<"\nEnter Keyword To Be updated :";

cin>>targetkey;

temp=avlsearch(root,targetkey);

if(temp==NULL)

{

cout<<"\nKeyword Is Not Present In The Dictionary..!!..";

}

else

{

cout<<"\n"<<temp->keyword<<" "<<temp->meaning;

cout<<"\nEnter New Meaning";

cin>>temp->meaning;

cout<<"\nYour Meaning has been Updated";

}

}

avlnode \* dictionary::avlsearch(avlnode \* temp,char targetkey[20])

{

if(temp==NULL)

{

return NULL;

}

if(strcmp(targetkey,temp->keyword)<0)

return avlsearch(temp->left,targetkey);

else if(strcmp(targetkey,temp->keyword)>0)

return avlsearch(temp->right,targetkey);

else

return temp;

}

int main()

{

dictionary dict;

int choice;

while(1)

{

cout<<"\n\n\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*";

cout<<"\n1. Add Keyword";

cout<<"\n2. Display Dictionary in Ascending order";

cout<<"\n3. Display in Descending Order";

cout<<"\n4. Search Keyword";

cout<<"\n5. Update Keyword In The Dictionary.";

cout<<"\n6. Exit";

cout<<"\n\nEnter your choice: ";

cin>>choice;

switch(choice)

{

case 1: dict.addkeyword();

break;

case 2: dict.display\_asc();

break;

case 3: dict.display\_dsc();

break;

case 4: dict.search\_keyword();

break;

case 5: dict.update\_keyword();

break;

case 6: exit(0);

default: cout<<"\n\nInvalid Choice: ";

}

}

}

**Output:**

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*

1. Add Keyword

2. Display Dictionary in Ascending order

3. Display in Descending Order

4. Search Keyword

5. Update Keyword In The Dictionary.

6. Exit

Enter your choice: 1

Enter no of Keywords : 2

Enter Keyword : were

Enter meaning : weare

Enter Keyword : go

Enter meaning : togo

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*

1. Add Keyword

2. Display Dictionary in Ascending order

3. Display in Descending Order

4. Search Keyword

5. Update Keyword In The Dictionary.

6. Exit

Enter your choice: 2

Keyword Meaning

---------------------------------

go togo

were weare

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*

1. Add Keyword

2. Display Dictionary in Ascending order

3. Display in Descending Order

4. Search Keyword

5. Update Keyword In The Dictionary.

6. Exit

Enter your choice: 3

Keyword Meaning

---------------------------------

were weare

go togo

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*

1. Add Keyword

2. Display Dictionary in Ascending order

3. Display in Descending Order

4. Search Keyword

5. Update Keyword In The Dictionary.

6. Exit

Enter your choice: 4

Enter Keyword To Be Searched :go

go togo

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*

1. Add Keyword

2. Display Dictionary in Ascending order

3. Display in Descending Order

4. Search Keyword

5. Update Keyword In The Dictionary.

6. Exit

Enter your choice: 5

Enter Keyword To Be updated :go

go togo

Enter New Meaninggogo

Your Meaning has been Updated

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*

1. Add Keyword

2. Display Dictionary in Ascending order

3. Display in Descending Order

4. Search Keyword

5. Update Keyword In The Dictionary.

6. Exit

Enter your choice: 6